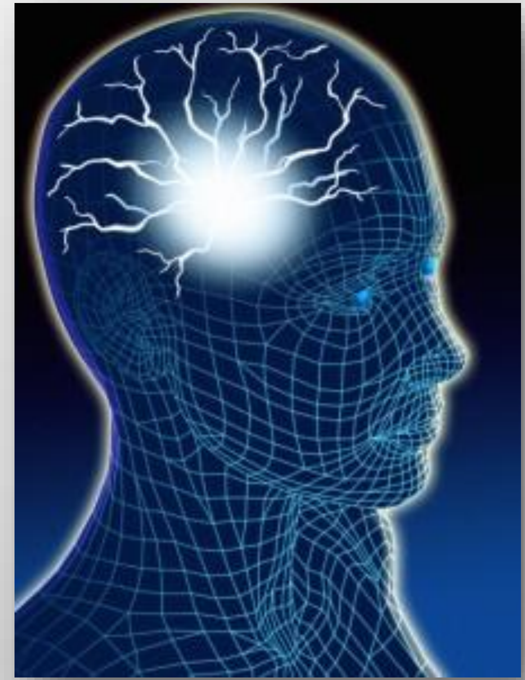
Solving problems by searching

Chapter 3



NORTH-WEST UNIVERSITY
YUNIBESITI YA BOKONE-BOPHIRIMA
NOORDWES-UNIVERSITEIT
INSTITUTIONAL OFFICE

Lecture outline

- Example problems
- Search Algorithms

Examples of problems

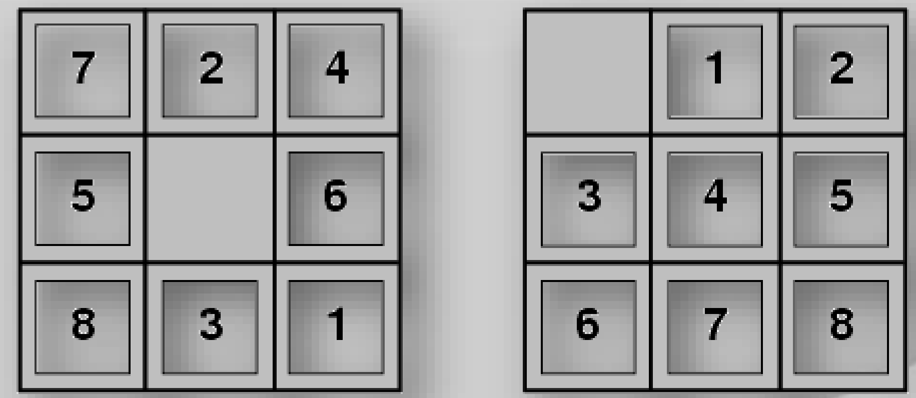
- Standardised problems
  - Intended to illustrate or exercise various problem solving methods
  - Can be given a concise, exact description
  - Is suitable as a benchmark for researchers to compare the performance of algorithms

Examples of problems

- Real world problems
  - Those whose solutions people actually use
  - Formulation is idiosyncratic, not standardised

Examples of problems

- The 8-puzzle (standardised problem)
  - States
  - Initial state
  - Actions
  - Transition model
  - Goal test
  - Path cost
- Abstractions?



Start State Goal State

Examples of problems

- The route-finding problem (real-world problem)
  - States
  - Initial state
  - Actions
  - Transition model
  - Goal test
  - Path cost

Examples of problems

- Other problems
  - Touring problems (TSP)
  - VLSI layout / PCB routing
  - Robot navigation
  - Automated assembly sequencing
  - Games
  - Etc.

Search Algorithms

- A search algorithm takes a search problem as input and returns a solution, or an indication of failure
- We consider algorithms that superimpose a search tree over the state-space graph, forming various paths from the initial state, trying to find a path that reaches a goal state
- Each node in the search tree corresponds to a state in the state space and the edges in the search tree correspond to actions

Search Algorithms

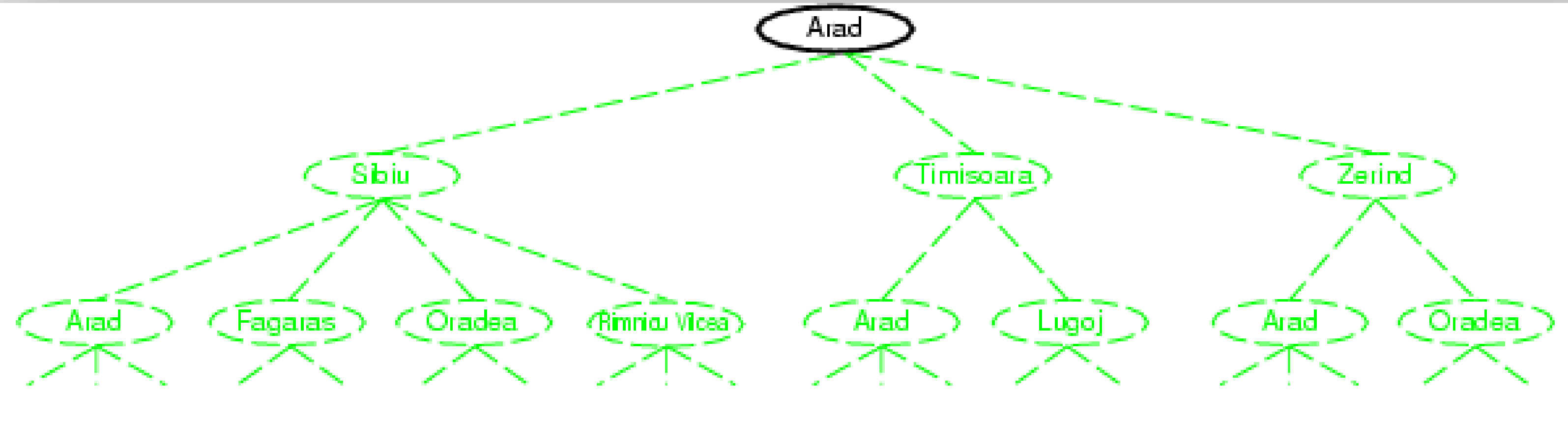
- The root of the tree corresponds to the initial state of the problem
- It is important to understand the distinction between the state space and the search tree
- The state space describes the (possibly infinite) set of states in the world, and the actions that allow transitions from one state to another

Search Algorithms

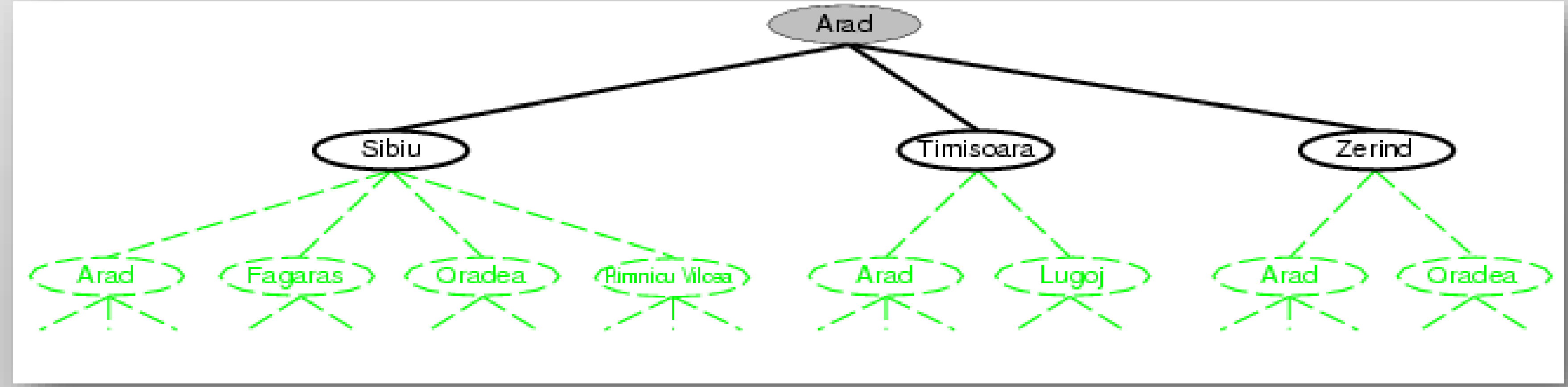
- The search tree describes paths between these states, reaching towards the goal
- The search tree may have multiple paths to (and thus multiple nodes for) any given state, but each node in the tree has a unique path back to the root (as in all trees)

Search Algorithms

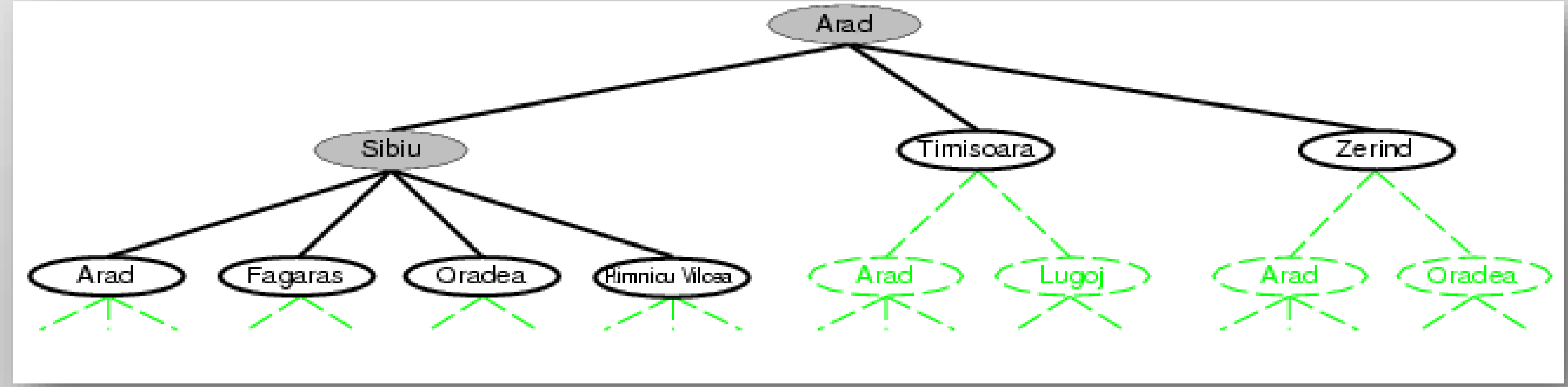
- Perform search through state space



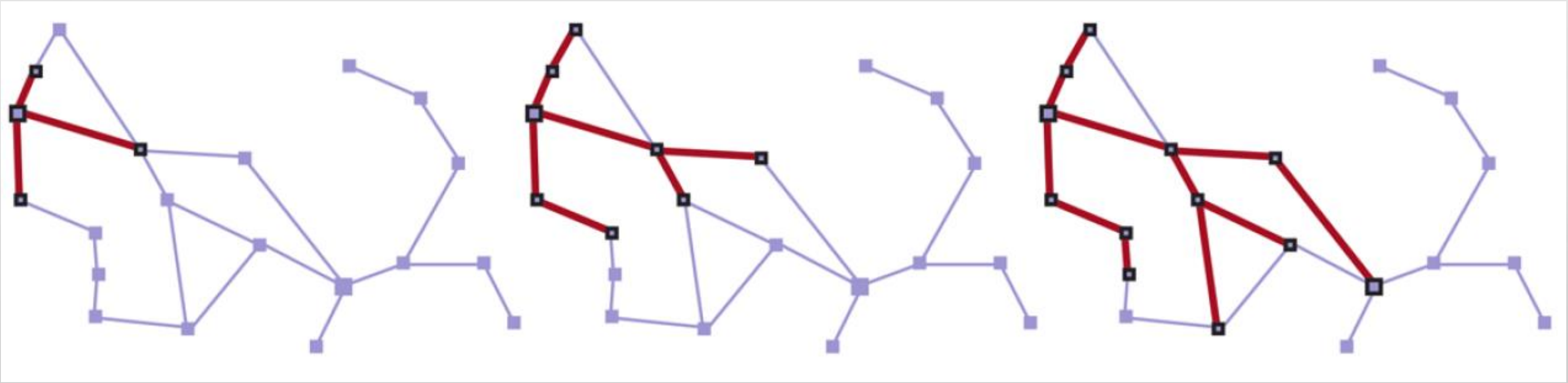
Search Algorithms



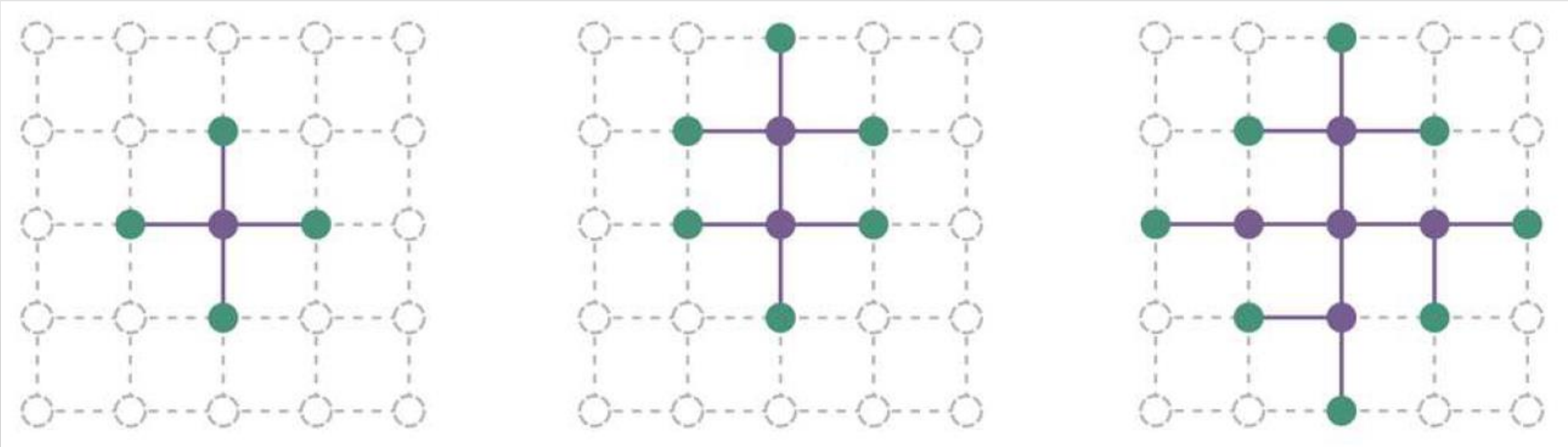
Search Algorithms



Search Algorithms



Search Algorithms



Best-first search

- How do we decide which node from the frontier to expand next?
- A very general approach is called best-first search, in which we choose a node, with minimum value of some evaluation function, *f(n)*
- On each iteration we choose a node on the frontier with minimum *f(n)* value, return it if its state is a goal state, and otherwise apply EXPAND to generate child nodes

NORTH-WEST UNIVERSITY
YUNIBESITI YA BOKONE-BOPHIRIMA
NOORDWES-UNIVERSITEIT
INSTITUTIONAL OFFICE

Best-first search

- Each child node is added to the frontier if it has not been reached before, or is re-added if it is now being reached with a path that has a lower path cost than any previous path
- The algorithm returns either an indication of failure, or a node that represents a path to a goal
- Different *f(n)* functions, result in different specific algorithms

NORTH-WEST UNIVERSITY
YUNIBESITI YA BOKONE-BOPHIRIMA
NOORDWES-UNIVERSITEIT
INSTITUTIONAL OFFICE

Search data structures

- A node in the tree is represented by a data structure with four components: node.STATE, node.PARENT, node.ACTION and node.PATH-COST
- We need a queue data structure to store the frontier: IS-EMPTY(frontier), POP(frontier), TOP(frontier) and ADD(node, frontier)
- Three kinds of queues are used in search algorithms: priority, FIFO, LIFO

Redundant paths

- The search tree shown in Figure 3.4 (bottom) includes a path from Arad to Sibiu and back to Arad again
- We say that Arad is a repeated state in the search tree, generated in this case by a cycle (also known as a loopy path)
- Even though the state space has only 20 states, the complete search tree is infinite because there is no limit to how often one can traverse a loop

Measuring problem-solving performance

- COMPLETENESS: Is the algorithm guaranteed to find a solution when there is one, and to correctly report failure when there is not?
- COST OPTIMALITY: Does it find a solution with the lowest path cost of all solutions?
- TIME COMPLEXITY: How long does it take to find a solution? This can be measured in seconds, or more abstractly by the number of states and actions considered

Measuring problem-solving performance

- SPACE COMPLEXITY: How much memory is needed to perform the search?
- Time and space complexity are considered with respect to some measure of the problem difficulty
- In theoretical computer science, the typical measure is the size of the state-space graph: $|V| + |E|$
- This is appropriate when the graph is an explicit data structure, such as the map of Romania

Measuring problem-solving performance

- In many AI problems, the graph is represented only implicitly by the initial state, actions, and transition model
- For an implicit state space, complexity can be measured in terms of ***d*** the depth or number of actions in an optimal solution; ***m*** the maximum number of actions in any path; and ***b*** the branching factor or number of successors of a node that need to be considered

Assignment

- Study: Chapter 3.2 (Example Problems) – 3.3 (Search Algorithms) of the AIMA e-book
- Theory Quiz 5: Chapter 3.2 (Example Problems) – 3.3 (Search Algorithms) of the AIMA e-book
  - 6 May 2021

NORTH-WEST UNIVERSITY
YUNIBESITI YA BOKONE-BOPHIRIMA
NOORDWES-UNIVERSITEIT
INSTITUTIONAL OFFICE